Sampling location generator

Project aim

The goal of this project was to create an easy-to-use interface for generating sampling locations, which meet certain criteria, within an area of interest. In particular, this was focused on locating points to sample peat depth to ensure both a) proportionate representation of the slope and elevation profile found in the wider area of interest and b) that each sample point was within a minimum distance of at least one other point.

Motivation

This was motivated by work from my dissertation involving using existing datasets of peat depth to fit models estimating peat depth over a wider area. Peat formation is largely influenced by topographic factors, and linear models based on slope and elevation as predictor variables have been shown to predict peat depth well in areas where slope is the dominant control on peat formation. However, in flatter areas, where the factors governing peat accumulation are more complex, geostatistical models, using spatial autocorrelation in the data to make predictions, have been shown to outperform linear models. Importantly, to realise the benefits of geostatistical modelling it is essential that sampled peat depth points are sufficiently close to one another to be spatially autocorrelated. It is also important that the range of slope and elevation values which are present in the wider area of interest are represented in the data used to fit the model.

The prevailing method for manual sampling of peat depth involves taking samples at a regular distance on a uniform grid. Unfortunately, the up-shot of this is that samples tend to be too far apart for the geostatistical model to perform. Contact with practitioners conducting this sampling suggests they would be open and willing to changing their standard sampling practices, but that they would require clear and easy to follow direction in how to best do this. An ideal tool would be able to utilise pre-existing peat depth sample datasets and suggest additional sampling to be carried out which would introduce auto-correlation into the data.

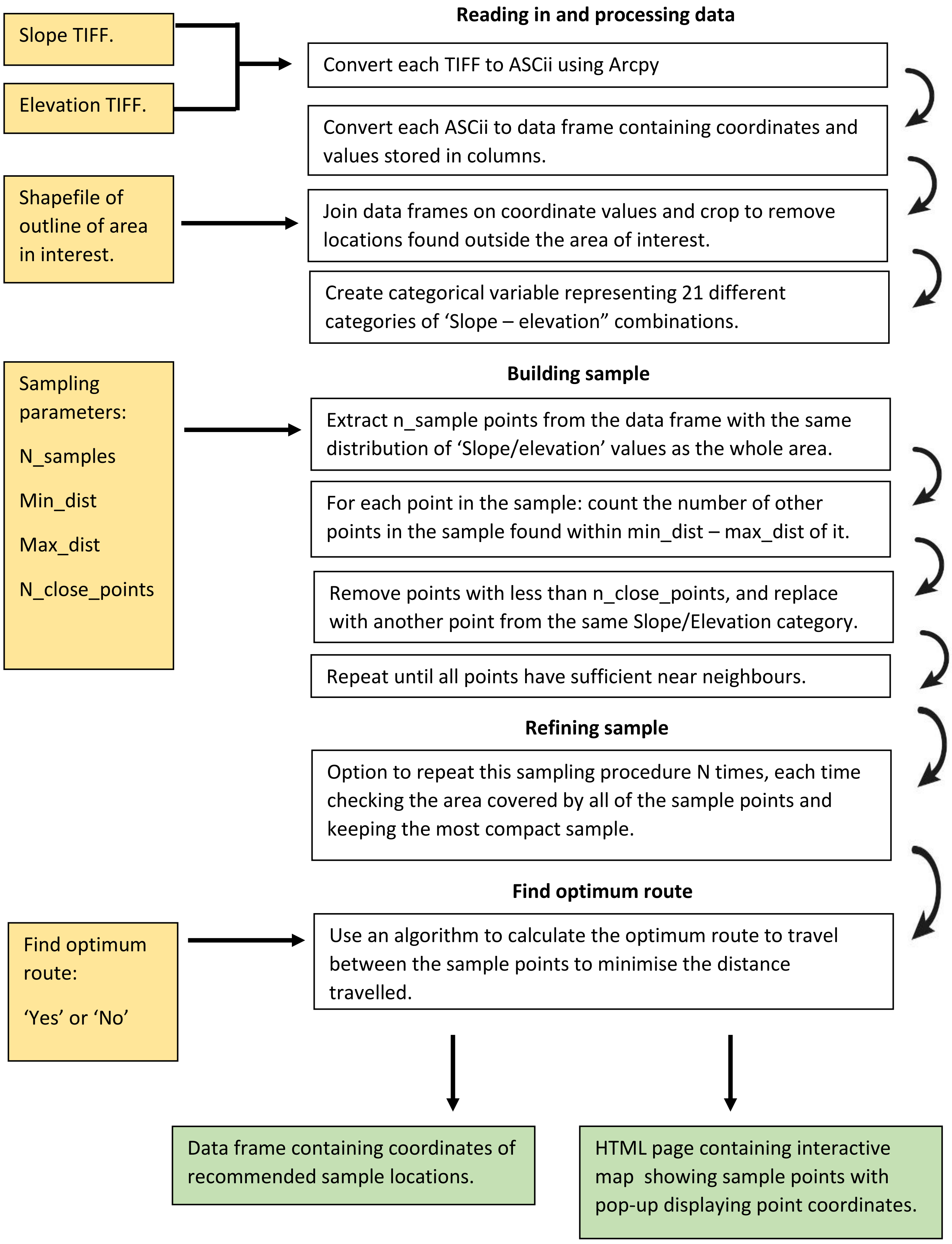
In considering this project I was particularly interested in developing skills for analysis of spatial data in Python and learning more about how to create an interactive user-interface.

Project workflow.

The final tool I created successfully implemented the work flow outlined in Figure 1. The project is structured with a config file to set-up the variables used in the RunProgram file, which does the main processing using functions defined in a separate function file.

Reflections

Reading in the raster data to Python proved to be more challenging than I had anticipated. Python's extension packages for processing raster data proved complicated to install on Windows. This conflicted with the project's main aim of producing an output which was readily useable by users with little programming experience. Thus, an alternative method of reading in the data, based on first using Arcpy to convert the TIFFs to ASCii files was devised. Although effective, this methodology is slower than implementation of functions in Python's 'Rasterio' module. Additionally, creating the code to manually read in the data exhausted a lot of the project time. The code development process could have been streamlined by creating a test dataset which was smaller and so quicker to test on.



Reading in and processing data
Slope TIFF.
Elevation TIFF.
Convert each TIFF to ASCii using Arcpy
Convert each ASCii to data frame containing coordinates and values stored in columns.
Shapefile of outline of area in interest.
Join data frames on coordinate values and crop to remove locations found outside the area of interest.
Create categorical variable representing 21 different categories of 'Slope – elevation" combinations.
Building sample
Sampling parameters:
N_samples
Min_dist
Max_dist
N_close_points
Extract n_sample points from the data frame with the same distribution of 'Slope/elevation' values as the whole area.
For each point in the sample: count the number of other points in the sample found within min_dist – max_dist of it.
Remove points with less than n_close_points, and replace with another point from the same Slope/Elevation category.
Repeat until all points have sufficient near neighbours.
Refining sample
Option to repeat this sampling procedure N times, each time checking the area covered by all of the sample points and keeping the most compact sample.
Find optimum route
Find optimum route:
'Yes' or 'No'
Use an algorithm to calculate the optimum route to travel between the sample points to minimise the distance travelled.
Data frame containing coordinates of recommended sample locations.
HTML page containing interactive map showing sample points with pop-up displaying point coordinates.

Figure 1: Main workflow for generating sampling locations according to slope and elevation within an area of interest.

Creation of a method to sample the data was also challenging. The final methodology iteratively removes points without sufficient near neighbours and randomly selects a point with the same slope/elevation characteristics to replace them. This method could be made more targeted, potentially be recording the ID of points whilst sampling to prevent resampling of the same point repeatedly. Despite this, I think it is a fairly simple method which resolves the problem quite effectively over relatively small timescales.

I added an additional stage to the sampling process which produces several versions of the sample and each time calculates the areal coverage of the sample points. The goal was to minimise the size of the area which had to be visited for sampling. However, this approach merely acknowledges the randomness in the sampling process and attempts to optimise the outcome through repetition. It might have been possible with more thought to introduce constraints to the sampling procedure itself which would have encouraged the points to migrate closer to each other. The explanation of this in the documentation could also be improved.

An option for calculating the optimum route between points to minimise the distance travelled to be calculated , using an algorithm produced by someone else based on the travelling salesman problem, was included. This was only really feasible for samples smaller than around 400 points, and is still very slow – and so is perhaps not that useful.

Figure 1 does not document an additional functionally I also attempted to implement (and for which the code has been included in a separate file which could be inserted into the main file as specified at line 94). This was intended to allow a user to import a shapefile containing locations in which they had already sampled the peat depth, and from this to identify additional locations they could take samples in to introduce spatial autocorrelation into their dataset. I tried to achieve this by:

- Adding a random extra sample of points to the existing sample
- Calculating the number of near neighbours of each point
- Resampling the dataset until each points has sufficient neighbours, with each iteration involving:
    - Firstly resampling all new points without enough near neighbours.
    - Once all new points have enough near neighbours, randomly replacing one of the original points with a new one

The code for this works as intended but the method does not seem appropriate, as in most cases too many of the original points are removed for the solution to be useful. This is a major area requiring further development. For instance, new points could be added without removing the original ones – as this confers no benefit. The travelling salesman (a.k.a travelling peat depth sampler) code used above contained some useful ideas for solving this problem which could be explored given more time.

Next steps

Due to the amount of time required to read in and process the data and sample I did not have time to attempt my initial aim to create a user interface. In the future I would like to attempt this using Bokeh. I envisage that this would involve the data reading and processing stages being conducted initially (as this is somewhat time consuming) and that then an interactive interface could be created which allows the user to change the number of samples, distances etc and visualise the locations of the samples produced. Despite this, the current project structure, which only requires a user to enter the config file, should still be relatively accessible to any user.